Computable combinatorial overfitting bounds*

Vorontsov K. V.[1], *Frey A. I.*[2], *Sokolov E. A.*[3]
voron@forecsys.ru, oleksandr.frei@gmail.com, sokolov.evg@gmail.com
[1]Dorodnitsyn Computing Centre, Russian Academy of Sciences; [2]Moscow Institute of Physics and Technology; [3]Moscow State University

In this paper we study computable combinatorial data dependent generalization bounds. This approach is based on simplified probabilistic assumptions: we assume that the instance space is finite, the labeling function is deterministic, and the loss function is binary. We use a random walk across a set of linear classifiers with low error rate to compute the bound efficiently. We provide experimental evidence to confirm that this approach leads to practical overfitting bounds in classification tasks.

Вычислимые комбинаторные оценки вероятности переобучения*

Воронцов К. В.[1], *Фрей А. И.*[2], *Соколов Е. А.*[3]
[1]Москва, Вычислительный Центр РАН; [2]Московский Физико-Технический Институт;
[3]Московский Государственный Университет

В данной статье изучаются комбинаторные оценки обобщающей способности, вычислимые по обучающей выборке. Эти оценки основаны на упрощенной вероятностной модели, в которой рассматривается лишь конечная генеральная совокупность объектов и бинарная функция потерь. Для линейных классификаторов предлагается новый эффективный метод вычисления комбинаторных оценок, использующий случайные блужданий по множеству классификаторов с низким числом ошибок. В заключении приводится экспериментальное обоснование предлагаемого метода.

Introduction

Accurate bounding of overfitting is an active area of research starting with the pioneer work [14]. A widely adapted approach is based on a probabilistic framework, where an instance space $\mathbb{X}$ (usually of an infinite cardinality) is equipped with an unknown probability distribution. Consider an observed i.i.d. sample $X = \{x_1, \ldots, x_\ell\}$ from $\mathbb{X}$, a set A of all feasible classifiers (for example, all linear classifiers in the original feature space), and a learning algorithm μ which selects a specific classifier $a = \mu X$ from the A based on the observed sample X. The goal of generalization bounds is to predict an average performance of the classifier a on the whole $\mathbb{X}$. Most generalization bounds are derived from various concentration inequalities [12] and take into account the dimensionality of A (such as VC-dimension, fat-shattering dimension, etc.), properties of the learning algorithm μ (such as the local properties of empirical risk minimization [8]), and information drawn from the observed sample X (such as the normalized margin in margin-based bounds [9, 10]). Generalization bounds can be useful in structural risk minimization and in model selection, and we hope that some time in the future they could replace costly cross-validation procedures.

*This work was supported by the Russian Foundation for Basic Research (project no. 11-07-00480, no. 12-07-33099-mol-a-ved) and by the program "Algebraic and Combinatorial Methods in Mathematical Cybernetics and New Generation Information Systems" of the Department of Mathematical Sciences of the Russian Academy of Sciences.

Despite recent significant improvements [5], there is still a big gap between theory and practice. Even the latest PAC-Bayesian bounds [7] vastly overestimate overfitting, especially when the cardinality of an instance space is small. Another difficulty is that intermediate bounds are usually expressed in terms of unobservable quantities and that makes impossible to measure and compare factors of overestimation. Finally, many papers lack experimental evaluation. As a result, from practical perspective the existing bounds are not well suitable for prediction and control of overfitting.

We believe that a simplified probabilistic framework is quite sufficient for obtaining practical overfitting bounds. In this paper we assume that the instance space $\mathbb{X}$ is finite, and for each object $x \in \mathbb{X}$ and classifier $a \in A$ there exists a deterministic binary loss $I(a, x) \in \{0, 1\}$ associated with classification of x as $a(x)$. We assume that all partitions of the set $\mathbb{X}$ into an observed training sample X of size ℓ and a hidden test sample $\bar{X} = \mathbb{X} \setminus X$ of size k can occur with equal probability. Then the overfitting probability can be defined by a purely combinatorial formula [15]:

$$Q_\varepsilon(\mu, \mathbb{X}) = \mathsf{P}\bigl[\nu(\mu(X), \bar{X}) - \nu(\mu(X), X) \geqslant \varepsilon\bigr], \tag{1}$$

where μ is a learning algorithm, $\nu(a, X)$ is an error rate of a classifier $a \in A$ on a sample X, and the square brackets denote a transformation of a logical value into a numerical one: $[\mathit{true}] = 1$, $[\mathit{false}] = 0$. A definition similar to (1) first appears in the [6] for a specific case of $k = 1$ and then in [4] (this time for an arbitrary k, but with significant notation differences). This definition closely resembles the procedure of complete cross-validation, which is known to provide sharp estimates of performance of a learning algorithm on data yet unknown during learning phase.

Definition (1) is not guarantied to be an upper bound for new objects beyond $\mathbb{X}$ (not even up to some probability). In this paper we do not discuss how to mitigate this problem. We just assume that the lack of guaranties is acceptable, in the same way as people normally accept results of a fare 10-fold cross-validation in experimental evaluations.

Within (1) we use an empirical risk minimization $\mu X = \arg\min\limits_{a \in A} \nu(a, X)$ as a learning algorithm. This requires an explicit representation of the set of classifiers A, which might be of enormous cardinality (10^9 and higher) in real applications. In this paper we study a special case of linear classifiers and present an efficient algorithm that samples a small set of classifiers (up to 10^4) to recover accurate overfitting bound (1). Also note that a direct computation of (1) is intractable because it involves a sum across all $\mathbb{C}_{\ell+k}^{\ell}$ subsets $X \subset \mathbb{X}$. For empirical risk minimization we use efficient upper bounds on (1), obtained in [16], and compare them with Monte-Carlo estimate of (1).

We study overfitting of logistic regression in experiments on 15 datasets from the UCI repository [2]. The results confirm that our approach provides sharp estimates of overfitting, which correlate with the actual overfitting, recovers a correct shape of the learning curves, and outperform the state-of-art PAC-bayesian bounds.

The rest of this paper is organized as follows. Section 2 contains a brief review of combinatorial bounds on (1). Section 3 describes our algorithm of sampling a representative set of linear classifiers. We provide experimental results in Section 4, and conclude in Section 5.

Background

Let $\mathbb{X} = \{x_1, \dots, x_L\}$ be a finite instance space and A be a set of classifiers. By $I\colon A \times \times X \to \{0, 1\}$ denote a binary loss function such that $I(a, x) = 1$ if a classifier a produces an error on an object x. For further consideration there is no need to specify what is “classifier”. Particularly, a regression function can also be a “classifier” if a binary loss function is used.

The binary vector $(a_i) \equiv \bigl(I(a, x_i)\bigr)_{i=1}^{L}$ of size L is called an *error vector* of the classifier a. Assume that all classifiers from A have pairwise different error vectors. The number of errors of a classifier a on a sample $X \subseteq \mathbb{X}$ is defined as $n(a, X) = \sum_{x\in X} I(a, X)$. The error rate is defined as $\nu(a, X) = \frac{1}{|X|} n(a, X)$. The subset $A_m = \{a \in A\colon n(a, \mathbb{X}) = m\}$ is called *m-layer* of classifiers.

A *learning algorithm* is a mapping $\mu\colon 2^{\mathbb{X}} \to A$ that takes a training sample $X \subseteq \mathbb{X}$ and gives a classifier $\mu X \in A$. The learning algorithm μ is called *empirical risk minimization* (ERM) whenever for all $X \in \mathbb{X}$ it satisfies $\mu X \in A(X)$, where

$$A(X) \equiv \operatorname{Arg}\min_{a\in A} n(a, X).$$

The choice of a classifier that minimizes empirical risk may be ambiguous because of discreteness of the function $n(a, X)$. ERM algorithm μ is said to be *pessimistic* if

$$\mu X \in \operatorname{Arg}\max_{a\in A(X)} n(a, \bar{X}).$$

The pessimistic ERM cannot be implemented in practice because it looks into a hidden part of data $\bar{X}$ unknown at the learning stage. Nevertheless, pessimistic ERM is a very useful theoretical concept because it gives tight upper bounds of overfitting probability for any ERM.

Permutational probability. By $[\mathbb{X}]^\ell$ denote a set of all $\mathbb{C}_L^\ell = \frac{L!}{\ell!(L-\ell)!}$ samples $X \subset \mathbb{X}$ of size ℓ. Define a probability operator P and an expectation operator E for a predicate $\varphi\colon [\mathbb{X}]^\ell \to \{0, 1\}$ and a real function $\psi\colon [\mathbb{X}]^\ell \to \mathbb{R}$:

$$\mathsf{P}\,\varphi = {\mathbb{C}_L^\ell}^{-1} \sum_{X\in[\mathbb{X}]^\ell} \varphi(X), \qquad \mathsf{E}\psi = {\mathbb{C}_L^\ell}^{-1} \sum_{X\in[\mathbb{X}]^\ell} \psi(X).$$

If the *discrepancy* $\delta(a, X) = \nu(a, \bar{X}) - \nu(a, X)$ is greater than a given nonnegative threshold ε, then the classifier $a = \mu X$ is said to be *overfitted*. Our goal is to estimate the *probability of overfitting*:

$$Q_\varepsilon(\mu, \mathbb{X}) = \mathsf{P}\bigl[\delta(\mu, X) \geqslant \varepsilon\bigr].$$

where $\delta(\mu, X) = \delta(\mu X, X)$ for short.

The *inversion* of an upper bound $Q_\varepsilon \leqslant \eta(\varepsilon)$ is an inequality $\nu(\mu X, \bar{X}) - \nu(\mu X, X) \leqslant \varepsilon(\eta)$ that holds with probability at least $1-\eta$, where $\varepsilon(\eta)$ is the inverse function for $\eta(\varepsilon)$. The *median of an upper bound $Q_\varepsilon \leqslant \eta(\varepsilon)$* is the inversion at $\eta = 1/2$.

Average train and test errors are defined as follows:

$$\nu_\ell(\mu, \mathbb{X}) = \mathsf{E}\nu(\mu X, X); \tag{2}$$
$$\bar{\nu}_\ell(\mu, \mathbb{X}) = \mathsf{E}\nu(\mu X, \bar{X}). \tag{3}$$

Hypergeometric distribution. For a classifier a such that $m = n(a, \mathbb{X})$ the probability to have s errors on a sample X is given by a hypergeometric function:

$$\mathsf{P}[n(a, X) = s] = \mathbb{C}_m^s \mathbb{C}_{L-m}^{\ell-s} {\mathbb{C}_L^\ell}^{-1} \equiv h_L^{\ell,\,m}(s)\,,$$

where argument s runs from $s_0 = \max\{0, m - k\}$ to $s_1 = \min\{m, \ell\}$, and parameter m takes values $0, \dots, L$. It is assumed that $\mathbb{C}_m^s = h_L^{\ell,\,m}(s) = 0$ for all other integers m, s.

Define the hypergeometric cumulative distribution function (left tail of the distribution):

$$H_L^{\ell,\,m}(z) = \sum_{s=s_0}^{\lfloor z\rfloor} h_L^{\ell,\,m}(s)\,.$$

Consider a set $A=\{a\}$ containing a fixed classifier so that $\mu X = a$ for any X. Then the probability of overfitting Q_ε transforms into the probability of large deviation between error rates on two samples $X, \bar X$. If the number of errors $n(a,\mathbb{X})$ is known, then an exact Q_ε bound can be obtained.

Theorem 1 (FC-bound [16]). *For a fixed classifier a such that $m=n(a,\mathbb{X})$, any set $\mathbb{X}$, and any $\varepsilon\in[0,1]$ the probability of overfitting is given by the left tail of the hypergeometric distribution:*

$$Q_\varepsilon(a,\mathbb{X}) = H_L^{\ell,\,m}\left(\tfrac{\ell}{L}(m-\varepsilon k)\right). \tag{4}$$

The hypergeometric distribution plays a fundamental role for combinatorial bounds. Together with union bound (4) provides an upper estimate of $Q_\varepsilon(\mu,\mathbb{X})$ that holds for any learning algorithm μ.

Theorem 2 (VC-type bound [16]). *For any set $\mathbb{X}$, any learning algorithm μ, and any $\varepsilon\in[0,1]$ the probability of overfitting is bounded by the sum of FC-bounds over the set A:*

$$Q_\varepsilon(\mu,\mathbb{X}) \leqslant \mathsf{P}\bigl[\max_{a\in A}\delta(a,X)\geqslant\varepsilon\bigr] \leqslant \sum_{a\in A} H_L^{\ell,\,m}\left(\tfrac{\ell}{L}(m-\varepsilon k)\right), \quad m=n(a,\mathbb{X}). \tag{5}$$

There are two reasons for looseness of (5). First, most classifiers in A are bad and should have vanishing probability to be obtained as a result of learning. Nevertheless, the uniform deviation bound ignores the learning algorithm μ. Second, similar classifiers share their contribution, which is ignored by union bound. Better bound should account for actual learning algorithm and similarity between classifiers.

Splitting and connectivity bounds Define an order relation on classifiers $a\leqslant b$ as a natural order over their error vectors: $a_i\leqslant b_i$ for all $i=1,\dots,L$. Define a metric on classifiers as a Hamming distance between error vectors: $\rho(a,b)=\sum_{i=1}^{L}|a_i-b_i|$.

Theorem 3 (SC-bound [16]). *If learning algorithm μ is pessimistic ERM, then for any $\varepsilon\in[0,1]$ the probability of overfitting is bounded by the weighted sum of FC-bounds over the set A:*

$$Q_\varepsilon(\mu,\mathbb{X}) \leqslant \sum_{a\in A} \mathbb{C}_{L-q-r}^{\ell-q}\,\mathbb{C}_L^{\ell\;-1} H_{L-q-r}^{\ell-q,\,m-r}\left(\tfrac{\ell}{L}(m-\varepsilon k)\right), \tag{6}$$

where $m=n(a,\mathbb{X})$, $q=q(a)$ is upper connectivity and $r=r(a)$ is inferiority of a classifier a:

$$q(a) = \#\{b\in A\colon\ a<b \text{ and } \rho(a,b)=1\};$$
$$r(a) = \#\{x\in\mathbb{X}\colon\ I(a,x)=1 \text{ and } \exists b\in A,\ \text{such that } b\leqslant a \text{ and } I(x,b)=0\};$$

where for any set S notation $\#S$ stands for cardinality of S.

SC-bound (6) turns into VC-type bound (5) when all $q(a)$ and $r(a)$ are set to zeros.

The weight $P_a=\mathbb{C}_{L-q-r}^{\ell-q}\mathbb{C}_L^{\ell\;-1}$ in sum (6) is an upper bound on the probability $\mathsf{P}[\mu X=a]$ to get a given classifier a as a result of learning. This quantity decreases exponentially as the connectivity $q(a)$ or the inferiority $r(a)$ increase. This implies that approximate calculation of

$Q_\varepsilon(\mu, \mathbb{X})$ requires knowledge not about the full set A, but only about few bottom layers of A. This fact motivates an algorithm presented in the next section.

Sampling linear classifiers

One has to deal with the set of all classifiers A to calculate bounds (5), (6), or to estimate $Q_\varepsilon(\mu, \mathbb{X})$ directly from definition (1). In this section we describe an efficient algorithm which samples a small set of classifiers (about 10^4) sufficient to recover accurate overfitting bound.

Consider binary classification problem with labels $y_i \in \{-1, +1\}$, $i = 1, \dots, L$ assigned to objects $x_i \in \mathbb{X} \subset \mathbb{R}^d$, respectively. Consider a set of unbiased linear classifiers $a(x; w) = = \operatorname{sign}\langle w, x\rangle$ where $w \in \mathbb{R}^d$ is a real vector of weights. We say that a pair of classifiers (w_1, w_2) are *neighbors* if their classification differs only by one object: $x \in \mathbb{X}$ such that $\operatorname{sign}(\langle w_1, x\rangle) \cdot \cdot \operatorname{sign}(\langle w_2, x\rangle) = -1$.

Our immediate goal is to find all or some of neighbors of a given classifier w_0. Then we will use this procedure to organize random walk on the graph $G = (A, E)$ where vertices correspond to classifiers in A, and edges connect neighbor classifiers.

Finding neighbor classifiers along specific direction. *Dual transformation* D maps a point $x \in \mathbb{R}^d$ into hyperplane $D(x) = \{w \in \mathbb{R}^d\colon \langle w, x\rangle = 0\}$, and maps a hyperplane $h = \{x \in \mathbb{R}^d\colon \langle w, x\rangle = 0\}$ into point $D(h) = w$. Applying dual transformation D to finite set of points $\mathbb{X} \subset \mathbb{R}^d$ produces a set of hyperplanes $\mathbb{H} \equiv \{D(x_i)\}_{i=1}^L$. Each hyperplane $h_i \in \mathbb{H}$ divides $\mathbb{R}^d$ into two half-spaces

$$h_i^+ = \{w \in \mathbb{R}^d\colon\ \operatorname{sign}\langle w, x_i\rangle = y_i\};$$
$$h_i^- = \{w \in \mathbb{R}^d\colon\ \operatorname{sign}\langle w, x_i\rangle = -y_i\}.$$

These half-spaces h_i^+ and h_i^- correspond to linear classifiers giving correct and incorrect answer on x_i respectively. So to find all classifiers with given error vector $I = (I_i)_{i=1}^L$, $I_i \in \{+, -\}$ where "+" corresponds to correct answer and "−" corresponds to incorrect, we just find the intersection of half-spaces $\bigcap_{i=1}^L h_i^{I_i}$. This intersection contains all linear classifiers with error vector I (and only them). So a set of hyperplanes $\mathbb{H}$ dissects $\mathbb{R}^d$ into convex polytopes called *cells*, and the partition itself is called *an arrangement of hyperplanes* [1]. It can be shown that finding neighbors of classifier $w_0 \in \mathbb{R}^d$ is equivalent to finding cells adjacent to the cell of w_0 in arrangement $\mathbb{H}$.

In order to find a neighbor of the classifier w_0 we select an arbitrary vector $u \in \mathbb{R}^d$ and consider a parametric set of classifiers $\{w_0 + tu\colon t \geqslant 0\}$. This set corresponds to a ray in the space of classifiers which starts from w_0 and goes along the direction of u. An intersection of this ray with hyperplane $h_i \in \mathbb{H}$ is defined by condition $\langle w_0 + tu, x_i\rangle = 0$, e.g. for $t_i = -\frac{\langle w_0, x_i\rangle}{\langle u, x_i\rangle}$. Let $t_{(1)}$ and $t_{(2)}$ be the first and the second smallest positive values from $\{t_i\}$, $i = 1, \dots, L$. Whenever $t_{(1)} \neq t_{(2)}$ we conclude that $w' = w_0 + \frac{1}{2}(t_{(1)} + t_{(2)})u$ defines an adjacent classifier along direction u.

Random walk on classifiers graph. Techniques of random walk [3, 13, 11] provide common approach to sample vertices from huge graphs. They are based on stationary distributions of Markov chains and have nice properties when the sample is large. Our goal is to get a small sample sufficient to estimate overfitting from (1), (5) or (6). In this paragraph we discuss how to organize such random walk on A based on procedure that finds a random neighbor for $w \in A$.

Our algorithm is given in listing 1. It is controlled by the desired number of classifiers n, maximal number of layer m, the number of concurrent walks N, and the probability p of transition towards classifier with higher number of errors. The computational complexity of this algorithm is $O(Ldn)$.

Algorithm 1 Random walk on classifiers graph

```
Require: starting point $w_0$; sample $\mathbb{X} \subset \mathbb{R}^d$; integer parameters $N$, $m$, $n$; float parameter
    $p \in (0, 1]$;
Ensure: set of classifiers $A$ with unique error vectors.
Initialize concurrent random walk: $v_i = w_0$, $i = 1, \dots, N$;
Create set $A := \varnothing$;
while $A.size() < n$
    for all $i \in 1, \dots, N$
        Find neighbor $v_i'$ of $v_i$ along random direction $u \in \mathbb{R}^d$;
        if $n(v_i', \mathbb{X}) > n(v_i, \mathbb{X})$ then
            with probability $(1 - p)$ continue;
        else if $n(v_i', \mathbb{X}) > n(w_0, \mathbb{X}) + m$ then
            continue;
        $v_i = v_i'$;
        A.add($v_i$);
return $A$
```

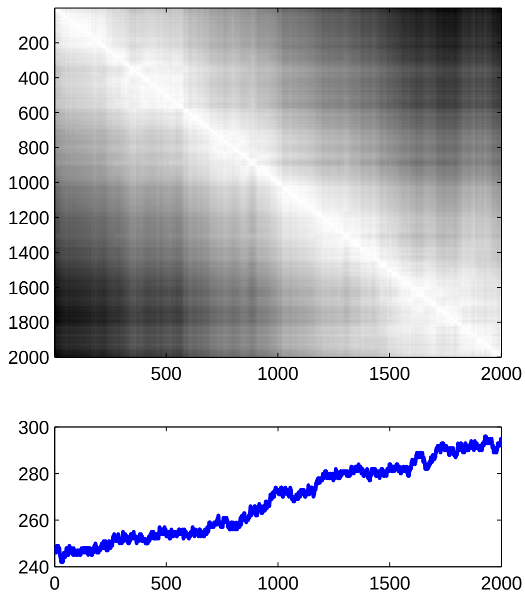

Fig. 1. Map of hamming distances between classifiers (top chart) and error profile (bottom chart) produced by a simple random walk

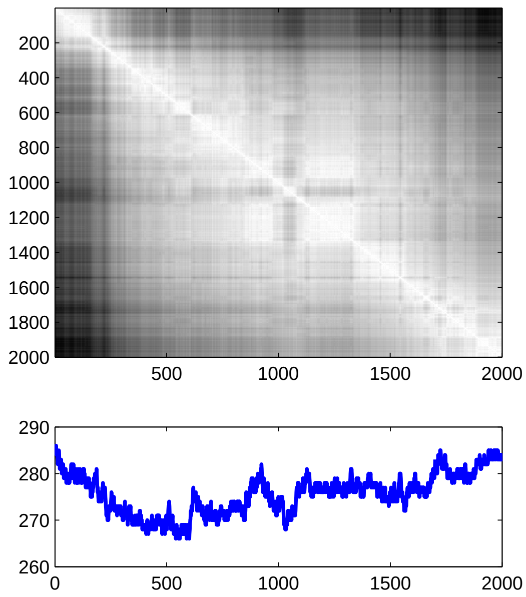

Fig. 2. Map of hamming distances between classifiers (top chart) and error profile (bottom chart) produced by random walk where a step to upper vertex is made with probability 0.5

To explain the necessity of the parameter p we presents the results of the simplest random walk with $n = 2000$ iterations on fig. 1. The bottom chart displays the number of errors $n(v_i, \mathbb{X})$ as a function of step. The upper chart displays a color map of pairwise hamming distances $\rho(v_i, v_j)$ between sampled vertices v_i and v_j. As a starting point we used a classifier learned by logistic regression. It is natural to expect that it has relatively small number of errors which drifts upwards along random walk. This effect is undesired, because classifiers with high number of errors have too small chance to be selected by learning algorithm.

Fig. 2 presents similar result for updated random walk where a step to upper vertex is made with probability $p = 0.5$. This enforces random walk to stay within the lower layers of the graph.

Experiment

The goal of our experiment on the benchmark datasets is twofold. First, we check whether combinatorial functionals $Q_\varepsilon(\mu, \mathbb{X})$ (1) and $\bar{\nu}_\ell(\mu, \mathbb{X})$ (3) together with algorithm 1 provide an accurate estimates of the overfitting on the hold-out testing sample. Second, we compare direct Monte-Carlo estimates of overfitting based on functional (1) with VC-type bound (5), SC-bound (6), and with recent PAC-bayesian DD-margin and DI-margin bounds proposed in [7].

Table 1. Description of datasets

Dataset	#Examples	#Features	Dataset	#Examples	#Features
Sonar	208	60	Statlog	2310	19
Glass	214	9	Wine	4898	11
Liver dis.	345	6	Waveform	5000	21
Ionosphere	351	34	Pageblocks	5473	10
Wdbc	569	30	Optdigits	5620	64
Australian	690	6	Pendigits	10992	16
Pima	768	8	Letter	20000	16
Faults	1941	27			

We use 15 datasets from the UCI repository [2]. If the dataset is a multiclass problem, we manually group the data into two classes since we study binary classification problem. For preprocessing we eliminate objects with one or more missing features and normalize all features into $[0, 1]$ interval. A description of the datasets is given in Table 1 with number of examples after elimination.

In all experiments we split the original dataset $\mathbb{X}$ into a training sample $\mathbb{X}_L$ and a testing sample $\mathbb{X}_K$. The training sample $\mathbb{X}_L$ is used to train a logistic regression and calculate overfitting bounds. Then we compare predictions of the bounds with the actual error rate on $\mathbb{X}_K$.

In the first experiment we build learning curves of logistic regression, where L runs from 5% to 95% of the original dataset size with 5% steps. For each L we generate $M = 100$ splits $\mathbb{X} = \mathbb{X}_L^i \cup \mathbb{X}_K^i,\ \ i = 1, \dots, M$ and use them to get Monte-Carlo estimates of train error rate $\nu_L(\mu_{\text{LR}}, \mathbb{X})$ from (2) and test error rate $\bar{\nu}_L(\mu_{\text{LR}}, \mathbb{X})$ from (3) for logistic regression learning algorithm μ_{LR}:

$$\hat{\nu}_L(\mu_{\text{LR}}, \mathbb{X}) = \frac{1}{M} \sum_{i=1}^{M} \nu(\mu_{\text{LR}} \mathbb{X}_L^i, \mathbb{X}_L^i), \qquad \hat{\bar{\nu}}_L(\mu_{\text{LR}}, \mathbb{X}) = \frac{1}{M} \sum_{i=1}^{M} \nu(\mu_{\text{LR}} \mathbb{X}_L^i, \mathbb{X}_K^i).$$

After that for each training sample $\mathbb{X}_L$ we sample classifiers and estimate an average ERM errors: train error $\nu_\ell(\mu, \mathbb{X}_L)$ and test error $\bar{\nu}_\ell(\mu, \mathbb{X}_L)$, where μ is ERM learning algorithm. To sample classifiers on $\mathbb{X}_L$ we launch algorithm 1 with parameters $n = 8\,192$, $N = 64$, $m = 15$, $p = 0.8$, and use classifier $\mu_{\text{LR}} \mathbb{X}_L$ as a starting point. To estimate $\nu_\ell(\mu, \mathbb{X}_L)$ and $\bar{\nu}_\ell(\mu, \mathbb{X}_L)$ we again compute Monte-Carlo type estimates of definitions (2) and (3) by randomly generating

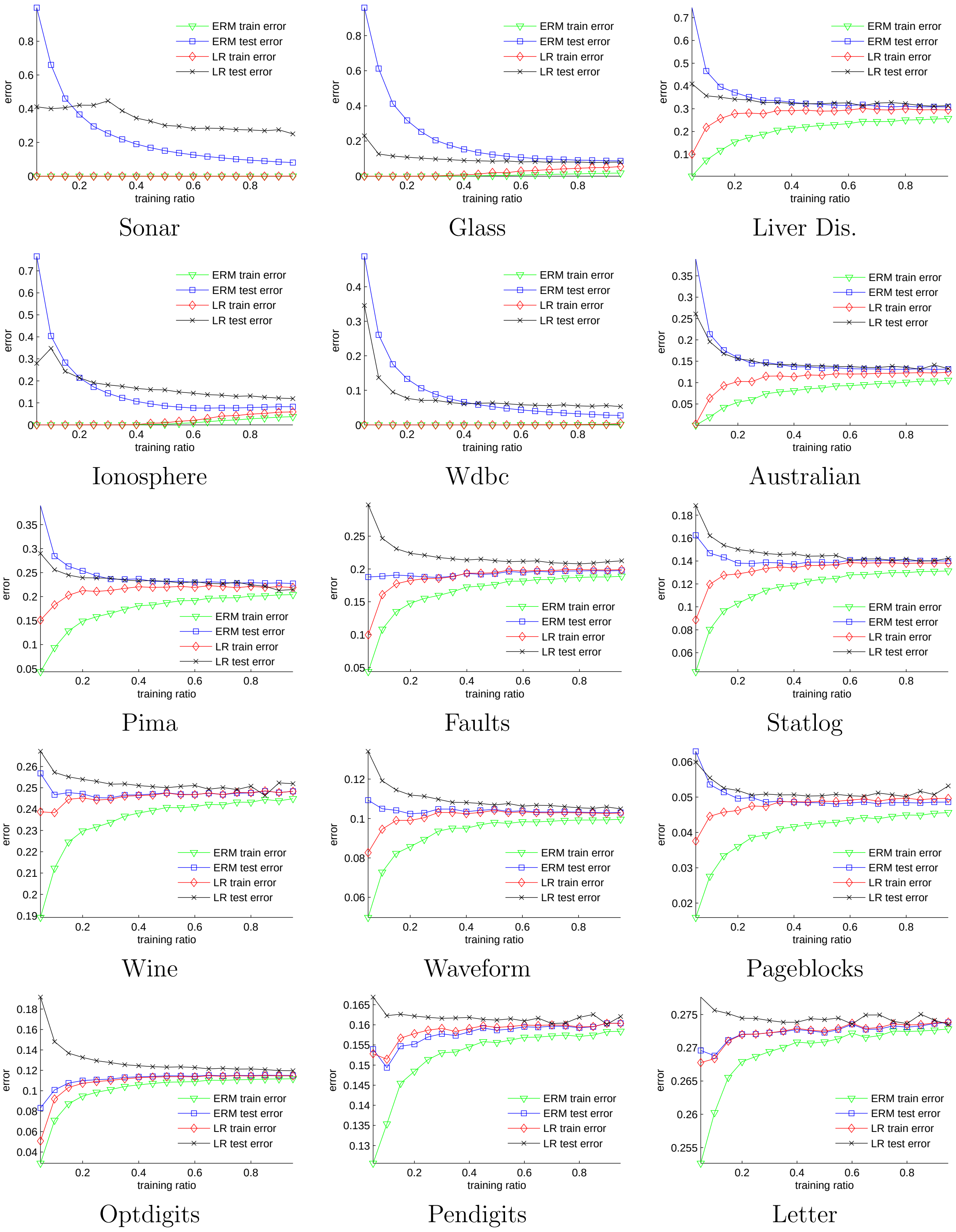

Fig. 3. Learning curves of logistic regression and ERM. The error ratio of logistic regression is estimated by Monte-Carlo method on splits of the original dataset $\mathbb{X} = \mathbb{X}_L \cup \mathbb{X}_K$. The error ratio of ERM is estimated on splits of the training set $\mathbb{X}_L = X_\ell \cup X_k$.

$M' = 4\,096$ splits $\mathbb{X}_L = X_\ell^j \cup X_k^j$, $j = 1, \dots, M'$, at constant ratio $\frac{\ell}{L} = 0.8$:

$$\hat{\nu}_\ell(\mu, \mathbb{X}_L) = \frac{1}{M'} \sum_{j=1}^{M} \nu(\mu X_\ell^j, X_\ell^j), \qquad \hat{\hat{\nu}}_\ell(\mu, \mathbb{X}_L) = \frac{1}{M'} \sum_{j=1}^{M} \nu(\mu X_\ell^j, X_k^j).$$

Table 2. Comparison between real overfitting and various overfitting bounds. TrainErr stands for $\nu_L(\mu_{\text{LR}}, \mathbb{X})$, TestErr for $\bar{\nu}_L(\mu_{\text{LR}}, \mathbb{X})$, Overfit is their difference, $\delta_\ell(\mu) \equiv \bar{\nu}_\ell(\mu, \mathbb{X}_L) - \nu_\ell(\mu, \mathbb{X}_L)$.

	Monte-Carlo estimates				Generalization bounds			
Task	TrainErr	TestErr	Overfit	$\delta_\ell(\mu)$	VC	SC	PAC DI	PAC DD
Sonar	0.000	0.271	0.271	0.095	0.185	0.119	1.287	1.287
Glass	0.046	0.075	0.029	0.078	0.211	0.140	1.126	0.738
Liver dis.	0.299	0.314	0.015	0.060	0.261	0.209	1.207	1.067
Ionosphere	0.049	0.125	0.077	0.052	0.150	0.112	1.219	1.153
Wdbc	0.001	0.056	0.055	0.032	0.071	0.043	1.174	0.705
Australian	0.122	0.136	0.013	0.030	0.137	0.110	1.146	0.678
Pima	0.220	0.227	0.007	0.028	0.159	0.127	0.971	0.749
Faults	0.198	0.210	0.012	0.010	0.108	0.087	1.110	1.061
Statlog	0.138	0.142	0.005	0.010	0.096	0.082	1.102	0.747
Wine	0.248	0.250	0.002	0.004	0.134	0.109	0.776	0.637
Waveform	0.103	0.105	0.002	0.004	0.099	0.079	0.561	0.354
Pageblocks	0.050	0.050	0.001	0.004	0.073	0.057	0.737	0.186
Optdigits	0.115	0.121	0.006	0.004	0.102	0.084	1.068	0.604
Pendigits	0.160	0.161	0.001	0.002	0.127	0.103	0.774	0.432
Letter	0.274	0.274	0.001	0.001	0.165	0.137	0.818	0.636

These estimates are then averaged over all partitions $\mathbb{X} = \mathbb{X}_L^i \cup \mathbb{X}_K^i$.

The four values (the actual train and test errors of logistic regression $\nu_L(\mu_{\text{LR}}, \mathbb{X})$ and $\bar{\nu}_L(\mu_{\text{LR}}, \mathbb{X})$, ERM train error $\nu_\ell(\mu, \mathbb{X}_L)$, and ERM test error $\bar{\nu}_\ell(\mu, \mathbb{X}_L)$) are charted as a functions of a training sample size ratio, see Figure 3, sorted according to sizes of datasets, from the smallest to the largest. Note that ERM test error might be either below or above actual test error rate of logistic regression because μ and μ_{LR} are quite different learning algorithms. However, from charts we conclude that $\bar{\nu}_\ell(\mu, \mathbb{X}_L)$, estimated only based on $\mathbb{X}_L$, provides reasonably good estimate of actual test error rate $\bar{\nu}_L(\mu_{\text{LR}}, \mathbb{X})$ and of the learning curve on test sample $\mathbb{X}_K$.

Now we turn to comparison of different overfitting bounds. For each dataset we use 5-fold cross validation and average the results over 20 runs (for a total 100 runs). As before, we use training sample $\mathbb{X}_L$ to learn logistic regression, run algorithm 1, and estimate $\nu_\ell(\mu, \mathbb{X}_L)$ and $\bar{\nu}_\ell(\mu, \mathbb{X}_L)$ based on 4 096 randomly generated splits $\mathbb{X}_L = X_\ell^j \cup X_k^j$. In addition, we use $\mathbb{X}_L$ to estimate overfitting $\bar{\nu}_\ell(\mu, \mathbb{X}_L) - \nu_\ell(\mu, \mathbb{X}_L)$ by medians of VC-type bound (5) and SC-bound (6), and to calculate DD-margin and DI-margin bounds from [7]. The results are presented in Table 2. Note that while all combinatorial bounds estimate overfitting, PAC DI and PAC DD are upper bounds on the test error.

Our key observation is that $\delta_\ell(\mu) \equiv \bar{\nu}_\ell(\mu, \mathbb{X}_L) - \nu_\ell(\mu, \mathbb{X}_L)$ is in the order of magnitude sharper than any of the other bounds. It works well for all datasets except Sonar (which is the smallest dataset in our selection). Across combinatorial bounds the SC-bound outperform VC-type bound, but still vastly overestimates the target quantity $\delta_\ell(\mu)$. All combinatorial bounds provide tighter estimates on overfitting and test error rate than PAC-Bayesian bounds.

Note that the VC-bound is estimated by a small subset of A obtained from a random walk. This is a “localized” VC-bound. The usual VC-bound estimated from VC-dimension d of a whole set A should be greater than 1 on all datasets.

Conclusion

In this paper we present new random walk technique for efficient calculation of combinatorial data-dependent generalization bounds. Although combinatorial bounds are obtained for empirical risk minimization under binary loss, we show that they provide sharp overfitting estimates for logistic regression. Our bounds recover correct shape of the learning curves for logistic regression, correlate well with the actual overfitting, and outperform both classical VC-bound and recent state-of-the-art PAC-Bayesian bounds in experiments on 15 datasets from the UCI repository.

References

[1] *Agarwal P. K., Sharir P.* Arrangements and their applications // In Handbook of Computational Geometry, pp. 49–119, 1998.

[2] *Asuncion A, Newman D. J.* UCI Machine Learning Repository // University of California, Irvine, School of Information and Computer Sciences, 2007.

[3] *Avrachenkov K., Ribeiro B., Towsley D. (2010)* Improving random walk estimation accuracy with uniform restarts // Proc. of WAW 2010.

[4] *Bax E.* Similar classifiers and VC error bounds, 1997.

[5] *Boucheron S., Bousquet O., Lugosi G.* Theory of classification: A survey of some recent advances // ESAIM: probability and statistics, vol. 9(1), pp. 323–375, 2005.

[6] *Haussler D., Littlestone N., Warmuth M. K.* Predicting $\{0,1\}$-functions on randomly drawn points // Information and Computation, vol. 115(2), pp. 248–292, 1994.

[7] *Jin C., Wang L.* Dimensionality Dependent PAC-Bayes Margin Bound // In Advances in Neural Information Processing Systems, vol. 25, pp. 1043–1051, 2012.

[8] *Koltchinskii V.* Local Rademacher complexities and oracle inequalities in risk minimization (with discussion) // The Annals of Statistics, vol. 34, pp. 2593-2706, 2006.

[9] *Koltchinskii V., Panchenko D.* Empirical margin distributions and bounding the generalization error of combined classifiers // The Annals of Statistics, vol. 30(1), pp. 1-50, 2002.

[10] *Koltchinskii V., Panchenko D.* Bounding the generalization error of convex combinations of classifiers: balancing the dimensionality and the margins // The Annals of Applied Probability, vol. 13(1), pp. 213-252, 2003.

[11] *Lee C., Xu X., Eun D.* Beyond Random Walk and Metropolis-Hastings Samplers: Why You Should Not Backtrack for Unbiased Graph Sampling // ACM SIGMETRICS Performance Evaluation Review, vol. 40(1), pp. 319–330, 2012.

[12] *Lugosi G.* On concentration-of-measure inequalities // Machine Learning Summer School, Australian National University, Canberra, 2003.

[13] *Ribeiro B., Towsley D.* Estimating and sampling graphs with multidimensional random walks // 10th Conf. on Internet Measurement, pp. 390–403, 2010.

[14] *Vapnik V. N., Chervonenkis A. Y.* On the uniform convergence of relative frequencies of events to their probabilities // Theory of Probability and Its Applications, vol. 16(2), pp. 264–280, 1971.

[15] *Vorontsov K. V.* Splitting and similarity phenomena in the sets of classifiers and their effect on the probability of overfitting // Pattern Recognition and Image Analysis, vol. 19(3), pp. 412–420, 2009.

[16] *Vorontsov K. V., Ivahnenko A. A.* Tight combinatorial generalization bounds for threshold conjunction rules. // 4-th Int'l Conf. on Pattern Recognition and Machine Intelligence (PReMI'11). Lecture Notes in Computer Science, Springer-Verlag, pp. 66–73, 2011.